

**Amendments to the claims,
Listing of all claims pursuant to 37 CFR 1.121(c)**

This listing of claims will replace all prior versions, and listings, of claims in the application:

What is claimed is:

1. (Original) In a computer system, an improved method for developing and executing an application, the method comprising:
 - creating a model describing business objects and rules of the application;
 - creating source code for the application, including representing the model within the source code itself;
 - compiling the source code into an executable application;
 - running the executable application on a target computer in conjunction with a run-time framework that provides services to the executable application; and
 - while the executable application is running, reconstructing the model from the executable application and making it available to the run-time framework.
2. (Original) The method of claim 1, wherein the model comprises a Unified Modeling Language (UML) model.
3. (Original) The method of claim 1, wherein the source code is created using a programming language.
4. (Currently amended) The method of claim 3, wherein the programming language is an object oriented programming languagea selected one of Java, Pascal, and C#.
5. (Original) The method of claim 3, wherein the programming language is one that supports reflection technique, thereby allowing reconstruction of the model at run-time from the executable application.
6. (Original) The method of claim 1, wherein the reconstructed model is

employed at run-time to support services that the run-time framework provides to the executable application.

7. (Original) The method of claim 1, wherein the reconstructing step includes:
using reflection, reading metadata associated with the executable application to
create a graph of code elements; and
spanning the graph for re-creating the model based on code elements encountered.

8. (Original) The method of claim 7, wherein the spanning step includes:
as each code element is encountered, reconstructing a corresponding portion of
the model.

9. (Currently amended) The method of claim 7, wherein the spanning step
includes traversing the graph using a selected one of depth-first, breadth-first, and ad-hoc
traversal techniques.

10. (Original) The method of claim 1, wherein the reconstructing step includes:
detecting a class having a package element; and
creating a corresponding Unified Modeling Language (UML) package for the
reconstructed model.

11. (Original) The method of claim 10, further comprising:
detecting an attribute specifying that a class belongs to the UML package; and
specifying in the reconstructed model that the class belongs to that UML package.

12. (Original) The method of claim 1, further comprising:
after reconstructing the model at run-time, testing integrity of the reconstructed
model.

13. (Original) The method of claim 12, further comprising:
ensuring that all classes in the model belong to a common superclass.

14. (Original) The method of claim 13, further comprising:
if all classes in the reconstructed model do not share a common superclass,
automatically constructing a common superclass for those classes.
15. (Original) The method of claim 1, wherein the reconstructed model is stored
in a cache memory available to the run-time framework.
16. (Original) The method of claim 1, wherein the model is initially created using
a modeling tool, and wherein the source code is compiled using a compiler.
17. (Currently amended) The method of claim 1, wherein the step of creating
source code includes:
representing ~~some~~ information of the model in source code as language
constructs.
18. (Currently amended) The method of claim 1, wherein the step of creating
source code includes:
representing ~~some~~ information of the model in source code as attributes.
19. (Original) The method of claim 18, wherein attributes comprise specifiers to
structural code elements.
20. (Currently amended) The method of claim 1, wherein the step of creating
source code includes:
representing ~~some~~ information of the model in code artifacts that exist expressly
for carrying model information in source code.
21. (Original) A computer-readable medium having processor-executable
instructions for performing the method of claim 1.

22. (Currently amended) A downloadable set of processor-executable instructions for performing the method of claim 1 stored on a computer readable medium.

23. (Original) In a computer system, an improved system for developing and executing an application, the system comprising:

a modeling tool for creating a model describing business objects and rules of the application;

a module for creating source code for the application and representing the model within the source code itself;

a compiler for compiling the source code into an executable application; and

a run-time framework that is able to reconstruct the model from the executable application and use it for providing services.

24. (Original) The system of claim 23, wherein the model comprises a Unified Modeling Language (UML) model.

25. (Original) The system of claim 23, wherein the source code is created using a programming language.

26. (Currently amended) The system of claim 25, wherein the programming language is an object oriented programming language a selected one of Java, Pascal, and C#.

27. (Original) The system of claim 25, wherein the programming language is one that supports reflection technique, thereby allowing reconstruction of the model at run-time from the executable application.

28. (Original) The system of claim 23, wherein the reconstructed model is employed at run-time to support services that the run-time framework provides to the executable application.

29. (Original) The system of claim 23, wherein the run-time framework includes submodules for reading metadata associated with the executable application to create a graph of code elements using reflection, and for spanning the graph for re-creating the model based on code elements encountered.

30. (Original) The system of claim 29, wherein the submodule for spanning is able to reconstruct portions of the model based on corresponding code elements encountered in the executable application.

31. (Currently amended) The system of claim 29, wherein the submodule for spanning is able to traverse the graph using a selected one of depth-first, breadth-first, and ad-hoc traversal techniques.

32. (Original) The system of claim 23, wherein the run-time framework includes submodules for detecting a class having a package element, and for creating a corresponding Unified Modeling Language (UML) package for the reconstructed model.

33. (Original) The system of claim 32, further comprising:
a module for detecting an attribute specifying that a class belongs to the UML package, and for specifying in the reconstructed model that the class belongs to that UML package.

34. (Original) The system of claim 23, further comprising:
a submodule for testing integrity of the reconstructed model.

35. (Original) The system of claim 34, further comprising:
a submodule for ensuring that all classes in the model belong to a common superclass.

36. (Original) The system of claim 35, further comprising:
a submodule for automatically constructing a common superclass for those classes

when all classes in the reconstructed model do not share a common superclass.

37. (Original) The system of claim 23, wherein the reconstructed model is stored in a cache memory available to the run-time framework.

38. (Original) The system of claim 23, wherein the model is initially created using a UML modeling tool, and wherein the source code is compiled using a C# compiler.

39. (Currently amended) The system of claim 23, wherein the module for creating source code is able to represent ~~some~~ information of the model in source code as language constructs.

40. (Currently amended) The system of claim 23, wherein the module for creating source code is able to represent ~~some~~ information of the model in source code as attributes.

41. (Original) The system of claim 40, wherein attributes comprise specifiers to structural code elements.

42. (Currently amended) The system of claim 23, wherein the module for creating source code is able to represent ~~some~~ information of the model in code artifacts that exist expressly for carrying model information in source code.

43. (Currently amended) A method for developing and executing an application on a computer system, the method comprising:

creating a model for developing an application using Unified Modeling Language (UML) technique;

generating source code to implement the model;

amending the source code for storing model information in the source code;

compiling the amended source code into an executable application and running

the executable application on the computer system;

reconstructing the model from the executable application; and

making the reconstructed model available for supporting operation of the executable application, including rendering the reconstructed model for display.

44. (Original) The method of claim 43, wherein the source code is implemented using a programming language.

45. (Currently amended) The method of claim 44, wherein the programming language is an object oriented programming language selected one of Java, Pascal, and C#.

46. (Original) The method of claim 45, wherein the programming language is one that supports reflection technique, thereby allowing reconstruction of the model from the executable application.

47. (Original) The method of claim 43, wherein the reconstructed model is employed by a run-time framework to provide services to the executable application.

48. (Original) The method of claim 43, wherein the reconstructing step includes:

using reflection, reading metadata associated with the executable application to create a graph of code elements; and

spanning the graph for re-creating the model based on code elements encountered.

49. (Original) The method of claim 48, wherein the spanning step includes:

as each code element is encountered, reconstructing a corresponding portion of the model.

50. (Currently amended) The method of claim 48, wherein the spanning step includes traversing the graph using a selected one of depth-first, breadth-first, and ad-hoc traversal techniques.

51. (Original) The method of claim 43, wherein the reconstructing step includes:
detecting a class having a package element; and
creating a corresponding Unified Modeling Language (UML) package for the
reconstructed model.

52. (Original) The method of claim 51, further comprising:
detecting an attribute specifying that a class belongs to the UML package; and
specifying in the reconstructed model that the class belongs to that UML package.

53. (Original) The method of claim 43, further comprising:
after reconstructing the model, testing integrity of the reconstructed model.

54. (Original) The method of claim 53, further comprising:
ensuring that all classes in the model belong to a common superclass.

55. (Original) The method of claim 54, further comprising:
if all classes in the reconstructed model do not share a common superclass,
automatically constructing a common superclass for those classes.

56. (Original) The method of claim 43, wherein the reconstructed model is stored
in a cache memory.

57. (Original) The method of claim 43, wherein the model is initially created
using a modeling tool, and wherein the amended source code is compiled using a
compiler.

58. (Currently amended) The method of claim 43, wherein the step of amending
the source code includes:

representing ~~some~~ information of the model in source code as language
constructs.

59. (Currently amended) The method of claim 43, wherein the step of amending the source code includes:

representing ~~some~~ information of the model in source code as attributes.

60. (Original) The method of claim 59, wherein attributes comprise specifiers to structural code elements.

61. (Currently amended) The method of claim 43, wherein the step of amending the source code includes:

representing ~~some~~ information of the model in code artifacts that exist expressly for carrying model information in source code.

62. (Original) A computer-readable medium having processor-executable instructions for performing the method of claim 43.

63. (Currently amended) A downloadable set of processor-executable instructions for performing the method of claim 43 stored on a computer readable medium.